Pending Claims:

1. A method for generating a dump file, the method comprising:

a. generating a dump file that does not include all operating system data by gathering at least:

i. thread information for at least one running thread,

ii. context information for the thread,

iii. callstack information for the thread,

iv. process information for a process in which the thread is running, and

v. information identifying a reason for generating the dump file; and

b. storing the dump file to a storage medium.

2. The method as recited in Claim 1, further comprising determining when to generate the dump file.

3. The method as recited in Claim 1, wherein generating the dump file further includes gathering processor information about at least one processor.

4. The method as recited in Claim 2, wherein determining when to generate the dump file further includes determining that an exception has occurred.

5. The method as recited in Claim 1, wherein the dump file does not include data stored in global initialized memory.

6. The method as recited in Claim 1, wherein the dump file does not include data stored in uninitialized memory.

7. The method as recited in Claim 1, wherein the dump file does not include executable instructions used by a processor to execute a program.

8. The method as recited in Claim 1, wherein the dump file is a kernel minidump file associated with an operating system and the at least one running thread is the single thread which encountered an exception.

9. The method as recited in Claim 8, wherein the callstack information includes kernel stack information.

10. The method as recited in Claim 1, wherein the process information identifies a process that initiated the thread.

11. The method as recited in Claim 1, further comprising:

allocating a buffer space in memory during an initialization process, wherein the buffer space is suitable for storing the gathered information; and

reserving space on the storage medium suitable for writing the contents of the buffer space.

12. The method as recited in Claim 11, wherein generating the dump file further includes initially storing the thread information, the context information, the callstack information, the process information, and the information identifying the reason for generating the dump file to the buffer space, and then copying the dump file from the buffer space to the storage medium as a minidump file.

13. The method as recited in Claim 12, further comprising upon re-initialization, after having stored the minidump file to the storage medium, accessing the minidump file on the storage medium and using at least a portion of the minidump file to further understand an exception that was at least one reason for generating the minidump file.

14. The method as recited in Claim 1, wherein the dump file is a user minidump file associated with at least one non-operating system program.

15. The method as recited in Claim 1, wherein generating the dump file further includes gathering callstack information for all running threads.

16. The method as recited in Claim 15, wherein the callstack information includes a user callstack.

17. The method as recited in Claim 1, wherein generating the dump file further includes gathering processor context information for all running threads.

18. The method as recited in Claim 1, wherein generating the dump file further includes gathering a listing of loaded modules for a faulting application program.

19. The method as recited in Claim 1, wherein the dump file is a directory indexed file that uses relative virtual addresses (RVAs).

20. A computer-readable medium having computer-executable instructions for causing at least one processor to perform acts comprising:

gathering dump file information that does not include all operating system data but does include at least thread information for at least one running thread, context information for the thread, callstack information for the thread, process information for the process in which the thread is running, and information identifying a reason for generating the dump file; and generating a dump file using the dump file information.

21. The computer-readable medium as recited in Claim 20, wherein generating the dump file further includes storing the dump file to a storage medium.

22. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering processor information about at least one processor.

23. The computer-readable medium as recited in Claim 20, having further computer-executable instructions for causing the at least one processor to perform acts comprising determining when to generate the dump file.

24. The computer-readable medium as recited in Claim 20, wherein the dump file does not include data stored in global initialized memory.

25. The computer-readable medium as recited in Claim 20, wherein the dump file does not include data stored in uninitialized memory.

26. The computer-readable medium as recited Claim 24 wherein the dump file does not include executable instructions used by the at least one processor to execute a program.

27. The computer-readable medium as recited in Claim 20, wherein the dump file is a kernel minidump file associated with an operating system and the at least one running thread is the single thread which encountered an exception.

28. The computer-readable medium as recited in Claim 20, wherein the callstack information includes kernel stack information.

29. The computer-readable medium as recited in Claim 20, wherein the process information identifies a process that initiated the thread.

30. The computer-readable medium as recited in Claim 20, further comprising computer-executable instructions for causing the at least one processor to perform acts comprising:

allocating a buffer space in memory during an initialization process, wherein the buffer space is suitable for storing the dump file information; and

reserving space on a storage medium drive suitable for writing the contents of the buffer space.

31. The computer-readable medium as recited in Claim 30, wherein generating the dump file further includes initially storing the thread information, the context information, the callstack information, the process information, and the information identifying the reason for generating the dump file to the buffer space, and then copying the dump file from the buffer space to the storage medium as a minidump file.

32. The computer-readable medium as recited in Claim 31, further comprising computer-executable instructions for causing the at least one processor to perform acts comprising, upon re-initialization after having stored the minidump file to the storage medium, accessing the minidump file on the storage medium and using at least a portion of the minidump file to further understand an exception that was at least one reason for generating the minidump file.

33. The computer-readable medium as recited in Claim 20, wherein the dump file is a user minidump file associated with at least one non-operating system program.

34. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering callstack information for all running threads.

35. The computer-readable medium as recited in Claim 34, wherein the callstack information includes a user callstack.

36. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering processor context information for all running threads.

37. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering a listing of all loaded modules for the faulting application program.

38. The computer-readable medium as recited in Claim 20, wherein the dump file is a directory indexed file that uses relative virtual addresses (RVAs).

39. An apparatus comprising;

memory;

a data storage drive configured to write data files to at least one data storage medium; and

at least one processor operatively coupled to the memory and the data storage drive and configured to:

a. generate a dump file that does not include all operating system data by gathering in the memory at least:

i. thread information for at least one running thread,

ii. context information for the thread,

iii. callstack information for the thread,

iv. process information for the process in which the thread is running, and

v. information identifying a reason for generating the dump file; and

b. store the dump file to the storage medium.

40. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to determine when to generate the dump file.

41. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to gather processor information about the at least one processor and include the processor information in the dump file.

42. The apparatus as recited in Claim 40, wherein the at least one processor is further configured to determining when to generate the dump file based on an exception.

43. The apparatus as recited in Claim 39, wherein the dump file does not include data stored in global initialized memory.

44. The apparatus as recited in Claim 39, wherein the dump file does not include data stored in uninitialized memory.

45. The apparatus as recited Claim 39 wherein the dump file does not include executable instructions used by the at least one processor to execute a program.

46. The apparatus as recited in Claim 39, wherein the dump file is a kernel minidump file associated with an operating system and the at least one running thread is the single thread which encountered an exception.

47. The apparatus as recited in Claim 39, wherein the callstack information includes kernel stack information.

48. The apparatus as recited in Claim 39, wherein the process information identifies a process that initiated the thread.

49. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to:

allocate a buffer space in the memory during an initialization process; and

reserve space on the storage medium drive suitable for writing the contents of the buffer space.

50. The apparatus as recited in Claim 49, wherein the at least one processor is further configured to:

generate the dump file by initially storing the thread information, the context information, the callstack information, the process information, and the information identifying the reason for generating the dump file to the buffer space, and then copying the dump file from the buffer space to the storage medium as a minidump file.

51. The apparatus as recited in Claim 50, wherein the at least one processor is further configured to, upon re-initialization after having stored the minidump file to the storage medium, access the minidump file on the storage medium and use at least a portion of the minidump file to further understand an exception that was at least one reason for generating the minidump file.

52. The apparatus as recited in Claim 39, wherein the dump file is a user minidump file associated with at least one non-operating system program.

53. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to gather callstack information for all running threads as part of the dump file.

54. The apparatus as recited in Claim 53, wherein the callstack information includes a user callstack.

55. The apparatus as recited in Claim 39, wherein the at least one processor is configured to gather processor context information for all running threads as part of the dump file.

56. The apparatus as recited in Claim 39, wherein the at least one processor is configured to gather a listing of all loaded modules for a faulting application program as part of the dump file.

57. The apparatus as recited in Claim 39, wherein the dump file is a directory indexed file that uses relative virtual addresses (RVAs).

67. The method as recited in Claim 1, further comprising providing the dump file to at least one external device.

68. The method as recited in Claim 12, upon system re-initialization, transferring the dump file from the storage medium to at least one external device.

69. The method as recited in Claim 1, wherein generating the dump file further includes gathering a list of loaded modules.

70. The computer-readable medium as recited in Claim 20, having further computer-executable instructions for causing the at least one processor to perform acts comprising providing the dump file to at least one external device.

71. The computer-readable medium as recited in Claim 30, having further computer-executable instructions for causing the at least one processor to perform acts comprising, upon system re-initialization, transferring the dump file from the storage medium to at least one external device.

72. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering a list of loaded modules.

73. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to provide the dump file to at least one external device.

74. The apparatus as recited in Claim 49, wherein the at least one processor is further configured to, upon system re-initialization, transferring the dump file from the storage medium to at least one external device.

75. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to gather a list of loaded modules as part of the dump file.